Amendments to the Specification:

Please replace the paragraph beginning at page 6, line 28, with the following rewritten paragraph:

As noted in U.S. Patent No. 5,694,224 in gray level printing, each pixel has the capability to be rendered in several different dot sizes or densities, and thus different gray levels. The number of gray levels is at least three whereas in a binary system only two levels are possible, background and highest density. However instead of simply providing each pixel with an independent gray level, several pixels may be organized together to form a super pixel, or cell. Each of the pixels in a cell is then provided with a gray level. The human visual response integrates the various gray levels of the individual pixels in the cell to a single perceived gray level for the cell. This is similar to the basic concept of binary halftoning. The number of tone ~~skills~~ levels for a cell is increased greatly, however, due to the number of different gray levels available for each pixel. For example, instead of only the two levels provided in binary halftoning for each pixel, 256 levels (including zero) can be provided with gray level printing for each pixel in the cell. The formation of the dots in the pixels of the cell can be performed in a number of different manners to achieve different desired results. The dots can be formed as "full" dot, "partial" dot, "mixed" dot or fixed dot to provide gray level halftoning. The partial dot formation process and mixed dot formation process are described in the aforementioned U.S. Patent No. 5,694,224.

Please replace the paragraph beginning at page 9, line 25, with the following rewritten paragraph:

A 1D (one dimensional) global color process control LUT 12 is used at the beginning to ensure the possibility of last-minute tuning of preference color even during the running of the printer after the images have already been RIPed (raster image processed). One input to LUT 12 is the 8 bit input data for the color separation image. The second input to LUT 12 is a color tweaking value for adjusting saturation of the color separation image. As shown in FIG. 18 there is provided a schematic illustrating the gray level input into LUT 12 and the corresponding gray level output from LUT 12 and the range of adjustments possible by modifying the color saturation on the output by the operator providing a color tweaking adjustment input that is available at the control panel of the workstation WS in FIG. 19. It will be noted that this input comes after the job image buffer and is effectively modifying image data after the image data is output from the job image buffer. Thus experimentation may be done by the operator in making copies (such as proof copies) with various tweaking adjustments without rescanning of original hardcopy documents or rerasterizing of the image data when the data is presented in electronic form as indicated in blocks 424 and 425 in FIG. 19. The job image buffer, JIB (424) buffers many ripped and compressed color separation files (C,M,Y,K), and pass a separation file page to the color separation page buffer (425) which decompresses the compressed image data and stores the page separation file for further processing, including real-time global color tweaking (see FIG. 1) based on last minute customer preference information from the WS. The yellow color separation page buffer and subsequent processing is shown in FIG.19 explicitly. The other equivalent color separation page buffers and subsequent processing for C, M and K channels are implied in FIG. 19. Preference color tweaking provides the last step of minor color adjustment to allow a user to adjust color if the user doesn't like the color being printed as may be observed from a proof print. Thus a de-saturated color may be adjusted back to a more saturated color. There may be provided the boosting of a specific color in the image. The coloring is not intended to provide fine-tuning of each color to be color accurate or to match color as a known color management process may be provided in a front end portion of the machine prior to rasterization. For full color or process color processes (cyan, magenta, yellow and optionally black) color tweaking is preferably performed before halftone processing because there are improved results obtained by modifying the contone (continuous tone) data rather than the halftone processed data. An advantage of having adjustments be provided to the contone data is that modifications to a dot structure or dot data formed after a halftone process may introduce unwanted artifacts (interaction from other color channels) in the dot structure and tends to provide more color variations or at least tends to be more difficult to predict/control adjustment of color.

Please replace the paragraph beginning at page 14, line 5, with the following rewritten paragraph:

a3

A robust implementation of this processing is indicated by the flowchart of FIG. 9 where the pixel having coordinates (x,y) is mapped to a certain location (I,J) 940 in a brick plane which location is then provided as one input to a halftone screen lookup table 970 that also has input to it the gray value g(x,y) of the pixel. The lookup table 970 stores rendered pixel values for halftone rendering of the image pixel g(x,y). In this example there are 241 x 255 rendering values in the LUT (brick width times number of brick planes). Further reduction of the size of the table can be made by recognizing that gray value 0 and 255 have I and J values that are irrelevant since in this example each pixel having a gray value of 0 and 255 is rendered at that respective value. In the flowchart of FIG. 9 the pixel image coordinate value x,y is input to a calculator that takes the value of the x-coordinate and adds that to a value of the y coordinate which has been first divided by the brick height and then multiplied by a brick offset value. This sum is then divided by the brick width wherein only the remainder is retained as the brick coordinate value for I. For example, where X=178,Y=1,Bh=1,Bs=177,and Bw=241, the calculation is made of adding 178 + (1/1)177=355, which is then divided by the brick width of 241 to yield a remainder I=114. The J coordinate value is determined by taking the y coordinate value in the image plane and dividing it by the brick height and retaining the remainder as the value for J. In this example for this screen the value of J is always zero, however, as noted above, some screens may have a brick height of two or more and so the J coordinate in the brick plane becomes essential to determine. Implementation of the brick coordinate calculator may be by software as processed by a computer or by a chip that is designed to perform this calculation. The calculation may be expressed by the formula:

$$I = (X +(Y/Bh) * Bs) \%Bw$$

wherein “%” indicates that a division operation is made wherein a remainder is determined. As noted above, Bh in certain situations is equal to one so the equation simplifies in such situation to:

$$I = (X + Y * Bs) \%Bw$$

Please replace the paragraph beginning at page 17, line 25, with the following rewritten paragraph:

In order to generate rendered screen values for a tile the various tile parameters 242 such as screen angle, lines per inch, number of gray levels per pixel are considered. In addition the nature of the dot driver 241 and dot type growth 243 pattern are also considered. An example of a dot driver is illustrated in Fig. 24 for a 16x16 dot size driver having a circular or spiral type of growth pattern wherein dots in a cell tend to grow from the center outwardly. Other types of dot drivers may be used and suited to other shapes of growth patterns such as growth along a line, or an ellipse. These factors may be input to a dot membership function generator 244, which considers cells within a tile and the contribution of neighborhood pixels within that tile. A screen profile builder 245 may then be used to determine the total gray level in the tile by summing of the exposure values at pixel locations that are not yet quantized. A screen profile quantizer 246 then quantizes the individual pixel rendered values so that these values can be expressed in a form of a whole number, for example, 0-255 in a system having an eight bits per pixel bit depth.